

SIN-FPGA – DESCRIPTION PAR SCHEMA



<u>Documents ressources:</u> **http://www.altera.com/literature/lit-index.html**
Introduction to Quartus II : intro_to_quartus2.pdf
Documentation QUARTUS II : quartusii_handbook.pdf
Documentation KIT DE2 : DE2_UserManuall.pdf
Mode d'emploi du simulateur : ModelSim Tutorial : modelsim_tutorial_ug.pdf
Data sheet : http://www.altera.com/literature/lit-cyc2.jsp
Langage VHDL : Le livre de J.Veiss et M.Meaudre, la version de 2007 est libre et téléchargeable ici : http://books.google.fr/books?id=AKoIOwjcqnUC

Logiciels :
ALTERA QUARTUS II et Mentor Graphic ModelSim :
https://www.altera.com/download/software/quartus-ii-we
https://www.altera.com/download/software/modelsim

Matériel : Carte ALTERA DE2 : http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html

1) *QUARTUS II : Développement par schéma*

a. Création du projet

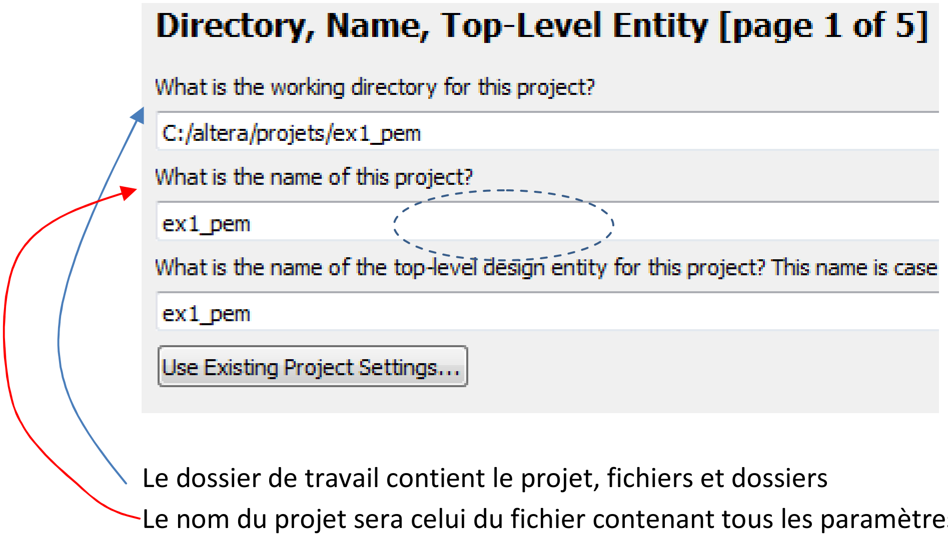
Les outils de développement pour FPGA utilisent la notion de projet :
- Définition de l'environnement du projet (Cibles, outils, contraintes et…)
- Travail collaboratif, plusieurs développeurs peuvent intégrer leurs productions dans un même environnement.

Ouvrir QUARTUS II V10.x, puis « Create un New Project »



Cliquer sur «New project wizard ». Le wizard (assistant) va nous guider dans la construction du projet.

Il est <u>indispensable</u> de structurer les projets dans des dossiers, l'outil QUARTUS générant un grand nombre de fichiers.



Le dossier de travail contient le projet, fichiers et dossiers
Le nom du projet sera celui du fichier contenant tous les paramètres du dossier

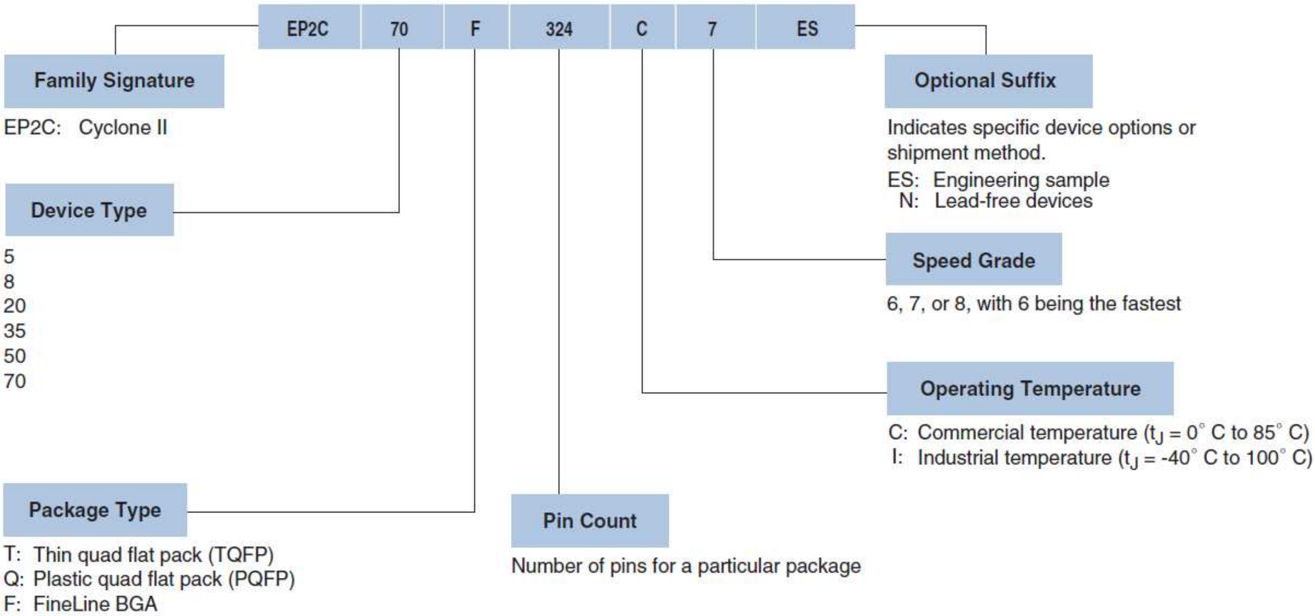
« Top-level design entity » est le nom du fichier de définition (schéma, MAE ou VHDL) de plus haut niveau, dans cet exemple il n'y a qu'un fichier schéma.

Compléter la page 1/5 du Wizard puis cliquer « next »



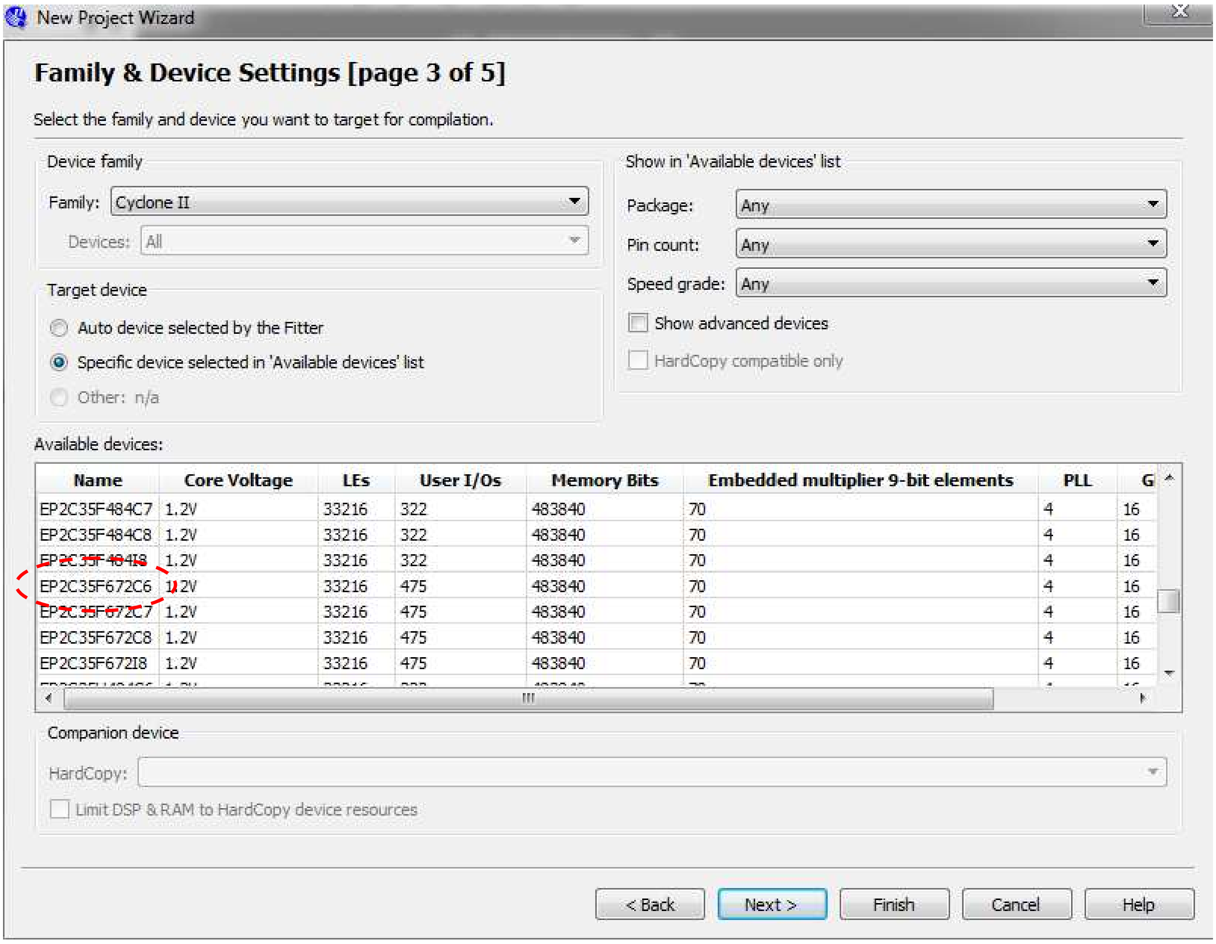
Page 2/5, « Add Files » permet d'ajouter des fichiers sources (schéma, MAE ou VHDL) au projet, ici il n'y en a pas. Cliquer « next » pour accéder à la page 3/5 du Wizard.

La page 3/5 permet de choisir le composant cible. La carte DE2 est équipée d'un FPGA Cyclone II, EP2C35F672C6 disposant de 33216 LEs (Logics Elements) en boitier BGA avec 672 broches, fonctionnant entre 0°C et 85°C et ayant un gradian de vitesse de 6 avec des broches soudables sans plomb. Le tableau ci-dessous explique le marquage du FPGA.



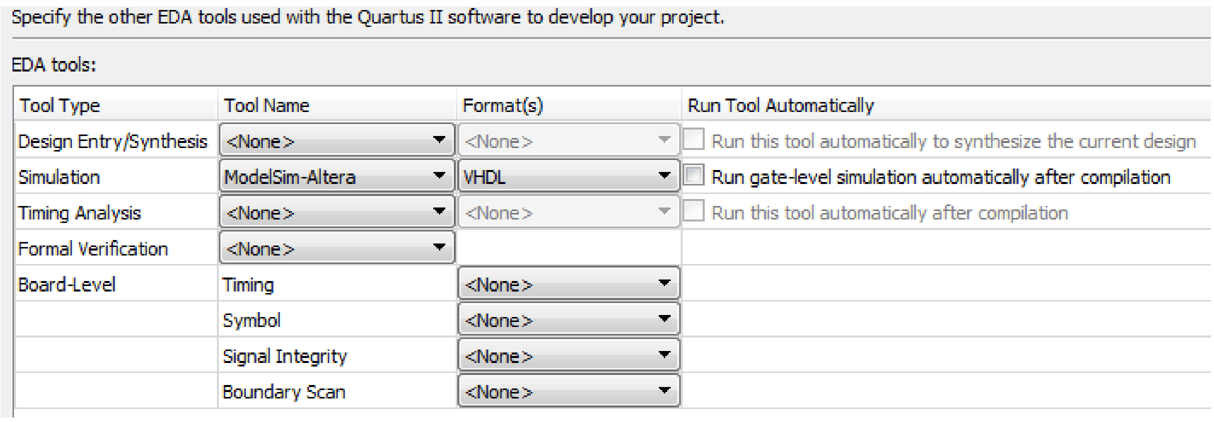
Ci-dessous, un exemple de comparaison des « speed grade », le FPGA équipant le KIT DE2 permet (entre autre) de synthétiser un multiplexeur 16 vers 1 fonctionnant à 259.87 MHz (le KIT DE2 est équipé d'un oscillateur 50 MHz)

Applications		Resources Used			Performance			
		LEs	M4K Memory Blocks	DSP Blocks	-6 Speed Grade	-7 Speed Grade	-8 Speed Grade	Units
LE	16-to-1 multiplexer *(3)*	11	0	0	382.99	302.84	259.87	MHz
	32-to-1 multiplexer *(3)*	24	0	0	292.74	237.98	204.49	MHz
	16-bit counter	16	0	0	445.23	387.74	341.64	MHz
	64-bit counter	64	0	0	188.82	168.37	150.92	MHz
Memory M4K block	Simple dual-port RAM 128 × 36 bit	0	1	0	260.01	216.73	180.57	MHz
	True dual-port RAM 128 × 18 bit	0	1	0	260.01	216.73	180.57	MHz
	FIFO 128 x 36 bit	24	1	0	260.01	216.73	180.57	MHz



Compléter la page 3/5 du Wizard puis cliquez « next »

ALTERA a choisi de ne pas intégrer de simulateur dans QUARTUS. Le simulateur ModelSim-ALTERA (société Mentor Graphics) est installé automatiquement avec QUARTUS, le développeur doit le préciser.



Compléter la page 4/5 du Wizard puis cliquer « next ».

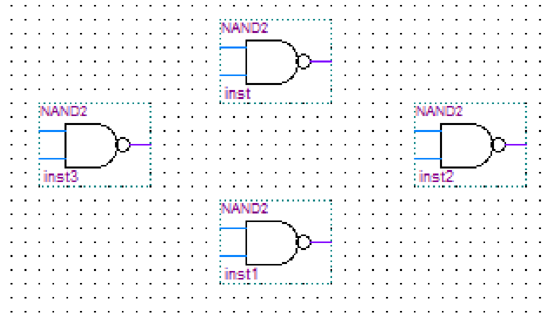
Vérifier la configuration sur la page 5/5 puis « Finish »
Le projet est créé et configuré, il est maintenant possible d'ajouter des fichiers de description schéma, MAE ou VHDL.

b. *Création d'un schéma*

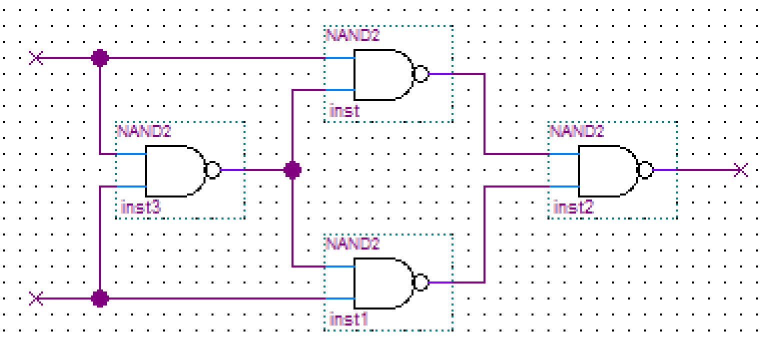
Cliquer « File – New » ou puis sélectionner « Block Diagram/schematic File », l'éditeur de schéma s'ouvre.

L'exercice propose la réalisation d'une fonction XOR à l'aide de quatre portes NAND (la fonction XOR existe)

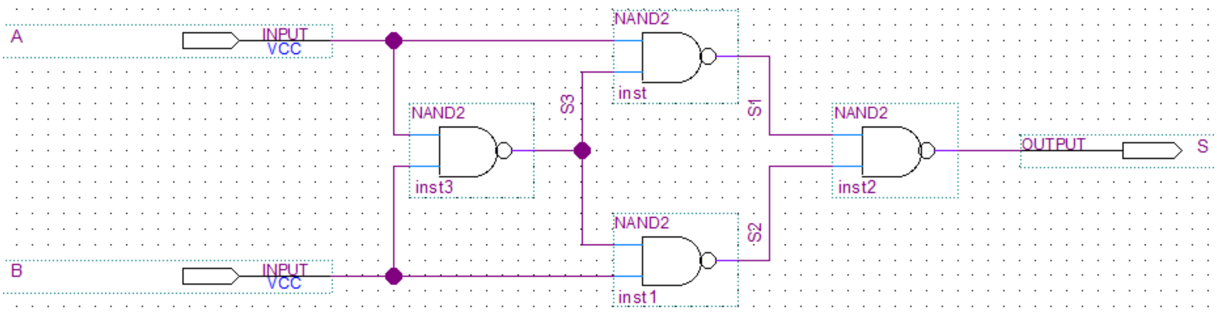
Cliquer puis choisir « primitives » «logic » et sélectionner une fonction NAND2. Placer quatre fonctions NAND comme ci-dessous.



Cliquer et réaliser ce schéma.

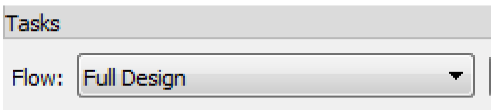


Cliquer et placer les entrées et sorties, nommer les (doubles clics pour éditer une connexion).
Nommer les signaux internes S1, S2, S3 (clic-droit sur le fil, puis properties)



Enregistrer le schéma ex1_pem.bdf en prenant soin de vérifier le dossier de destination (celui du projet)

Vérifier que la compilation est en mode « full design »



(la boite de selection « Tasks » est située sur la gauche de l'écran)

Cliquer pour compiler le projet. (Compter 30 secondes de compilation)

QUARTUS vérifie le schéma, crée un fichier VHDL correspondant, puis effectue un routage dans le FPGA cible. Il crée des rapports pour toutes les étapes de la compilation. Si tout est correct, La fenêtre de messages « Processing » indique : 0 errors, 9 warnings .

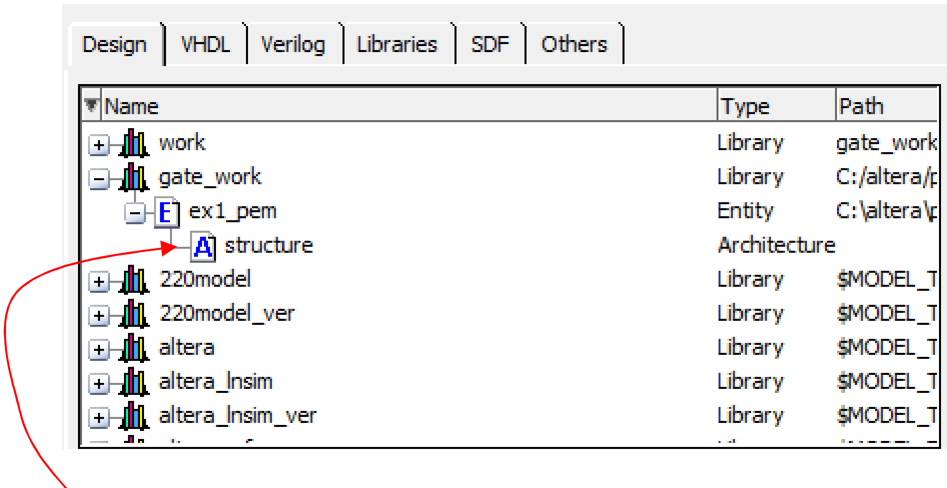
Les « warnings » indiquent que le compilateur a dû prendre des décisions qui peuvent influer sur le résultat attendu ou que la description (design) est peut être incomplète. Ici par exemple aucune horloge n'est utilisée. Généralement (!) les « warnings » peuvent être ignorés.
Le test du projet peut être effectué par le simulateur ModelSim.

c. Simulation

EDA : Electronic design automation (EDA or ECAD) ou Conception Assitée par Ordinateur en électroniqueCAO electronique
🖰 Cliquer « Tools – Run EDA simulation tools – EDA Gate Level Simulation “ , choisir “slow model”
Ce qui va lancer le simulateur en mode “gate level”, dans ce mode les caractéristiques temporelles du composant peuvent être prises en compte dans la simulation.

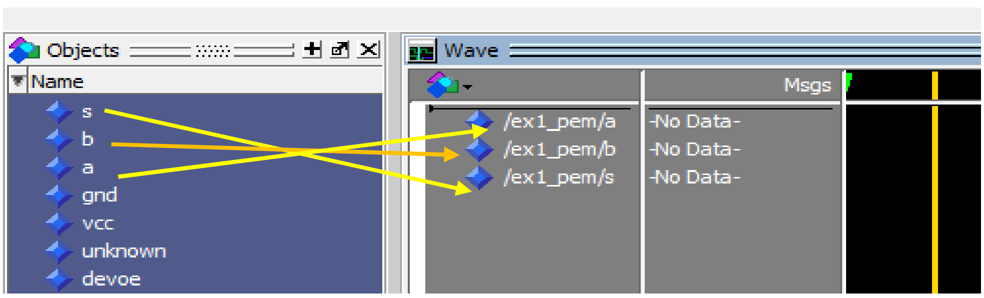
Si l'erreur "Can't launch the ModelSim-Altera software -- the path..." apparait, le chemin du simulateur, n'est pas correctement renseigné.
Pour corriger le problème, fermer la boite de dialogue de l'erreur windows aller dans Tools puis Option et choisir EDA Tool Options sous la catégorie General.
Dans l'espace ModelSim-Altera parcourir les fichiers pour pointer vers C:/altera/10.0/modelsim_ase/win32aloem" Valider par OK

(Le mode RTL : register transfer level, permet une simulation fonctionnelle à partir du (des) fichier VHDL de la description, ce mode ne prend pas en compte les temps de propagation dans le FPGA cible et sera présenté dans le paragraphe suivant)
L'interface QUARTUS-ModelSim compile automatiquement la description avec les caractéristiques temporelle du FPGA.
Dans ModelSim, 🖰 Cliquer “Simulate – Start Simulation”



Onglet “Design”, selectionner “structure”
Onglet “SDF” - ADD – browse , choisir “ex1_pem_vhd.sdo” puis “OK” (le fichier sdo est crée par QUARTUS lors de l'appel du simulateur, il contient les caractéristiques temporelles du FPGA.

Le simulateur propose dans la fenêtre « Objects » les « éléments » constituants la description, sélectionner a,b,s puis clic-droit « add to wave » ou « tirer » la sélection dans la fenêtre Wave.

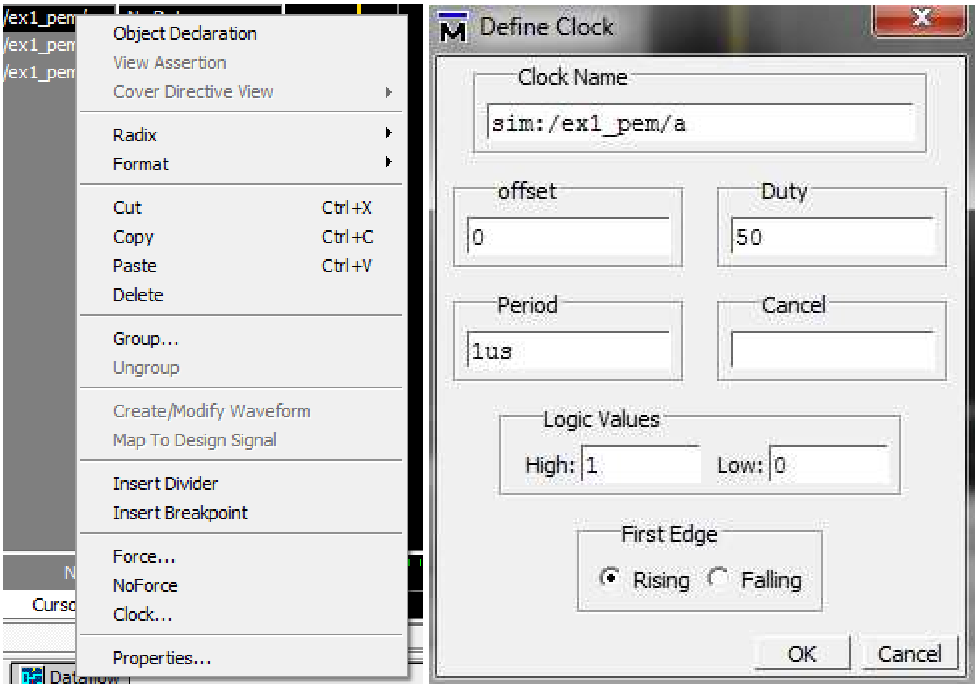


Il est possible de déplacer les signaux simplement avec la souris, afin de les visualiser dans un ordre particulier

De nouveaux signaux ont été créés par Quartus lors de la synthèse, ils représentent toutes les équipotentielles utiliseés par la description et synthétisées dans le FPGA. Le dessin d'origine n'est pas forcément respecté, Quartus optimisant la conception en place et en vitesse. ***Les signaux internes S1,S2 et S3 ne sont pas accessibles.***

Création des stimuli sur les entrées a et b

Cliquer- droit sur "a" puis Clock et créer une horloge avec une période de 1us et un rapport cyclique de 50%.

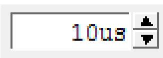


Faire de même pour « b » avec une période de 1.5us

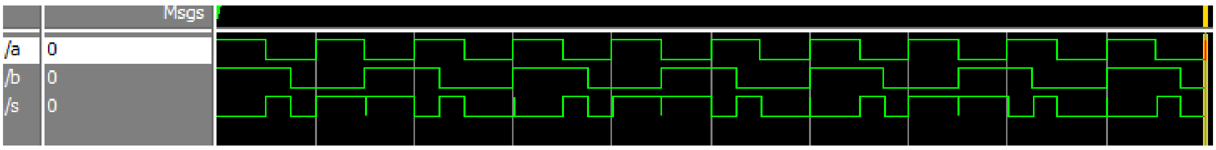
Noter dans la fenêtre « Transcript » les instructions engendrées

```
force -freeze sim:/ex1_pem/a 1 0, 0 {500000 ps} -r 1us
force -freeze sim:/ex1_pem/b 1 0, 0 {750000 ps} -r 1.5us
```

Ces instructions peuvent être écrites dans un fichier (avec l'extension « .do ») afin ***d'automatiser la simulation. Le lancement s'effectue alors avec la commande « do » ex : do test.do***

Ecrire 10us 

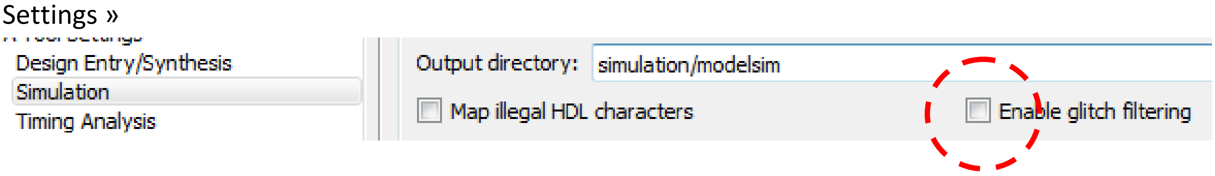
comme durée de simulation puis cliquer sur run  le simulateur déroule 10us de simulation. Cliquer dans la fenêtre de simulation puis  pour visualiser l'ensemble.

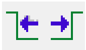


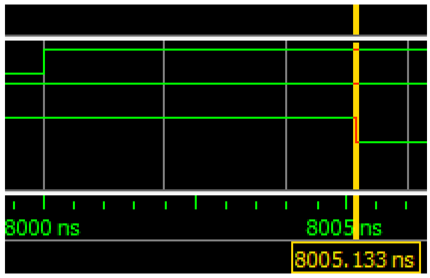
Vérifier la fonction « XOR » en déplaçant le curseur jaune sur les chronogrammes.

Le simulateur montre les « glitchs » dus aux temps de propagation dans le FPGA.

La visualisation des glitchs peut être exclue <u>dans le gestionnaire de projet QUARTUS</u>, « Assignments-Settings »



A l'aide de  positionner le curseur jaune sur une transition du signal de sortie « s » (pas un glitch) puis zoomer avec le bouton zoom situé à droite qui gardera le curseur au centre de la simulation.



On peut mesurer le temps de propagation depuis front montant sur « a » vers la sortie « s ». (ici 5,133ns), la fonction XOR supporterait une fréquence maximum de 194Mhz.

<u>Liste des commandes utilisables dans un fichier .do :</u>

Commande	Explication
restart -force	Réinitialise toute la simulation
destroy .wave	Ferme la fenêtre de simulation
add wave /* ou add wave -r /*	Ajoute un signal à la simulation. * ajoute tous les signaux (add horloge ajoute le signal « horloge » -r est utilisé lorsque plusieurs blocs sont simulés en même temps et qu'il est utile d' analyser tous les signaux de chaque bloc.
view wave	Ouvre la fenêtre de simulation graphique
force entree 0	Force un signal à un niveau logique. Ici entree est forcé à 0
force clk 0 0ns, 1 50ns -repeat 100ns	Création d'un signal de type horloge. 0 0 met clk à 0 à l'instant 0ns 1 50ns met clk à 1 à l'instant 50ns repeat 100ns répète la séquence précédente toutes les 100ns
#comment	Un commentaire
run 100ns	Lance le simulateur durant 100ns

Quitter ModelSim.

d. Simulation fonctionnelle

Il est souvent indispensable d'effectuer une simulation fonctionnelle du projet. ModelSim permet de simuler une description en langage de haut niveau (VHDL). Il est donc nécessaire de créer un fichier VHDL à partir du schéma (ou de tout autre moyen de description)

Génération du fichier VHDL à partir de la description par schéma.

Cliquer « File – Create/Update – Create HDL design from current file », choisir VHDL. Quartus crée un fichier VHDL à partir du schéma. Ce fichier est maintenant accessible et éditable depuis l'explorateur de projet de Quartus.

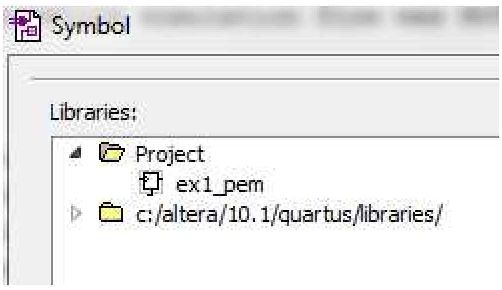
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY work;

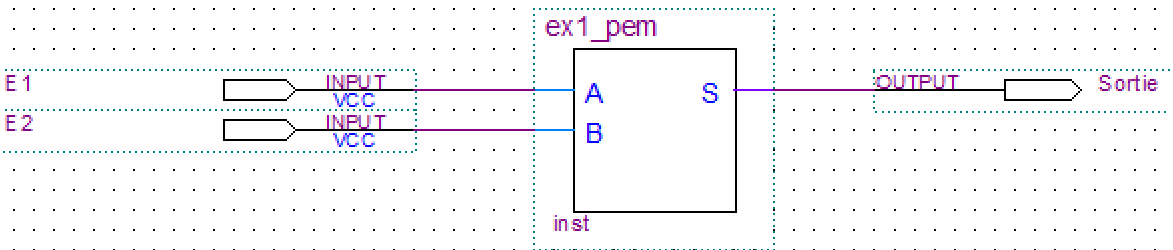
ENTITY ex1_pem IS
      PORT
      (
            A :  IN  STD_LOGIC;
            B :  IN  STD_LOGIC;
            S :  OUT  STD_LOGIC
      );
END ex1_pem;

ARCHITECTURE bdf_type OF ex1_pem IS
SIGNAL S_ALTERA_SYNTHESIZED :  STD_LOGIC_VECTOR(3 DOWNTO 1);
BEGIN
  S_ALTERA_SYNTHESIZED(1) <= NOT(S_ALTERA_SYNTHESIZED(3) AND A);
  S_ALTERA_SYNTHESIZED(2) <= NOT(B AND S_ALTERA_SYNTHESIZED(3));
  S <= NOT(S_ALTERA_SYNTHESIZED(2) AND S_ALTERA_SYNTHESIZED(1));
  S_ALTERA_SYNTHESIZED(3) <= NOT(B AND A);
END bdf_type;
```

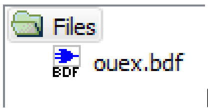
Ce fichier peut lui-même être transformé en un nouveau composant qui pourra être inséré dans un schéma, cela permet de conserver la possibilité de synthèse par schéma dans le FPGA.
Depuis la fenêtre d'édition du fichier VHDL :
Cliquer « File – Create/Update – Create symbol files for current file »
Le fichier ex1_pem.bsf est créé. (bsf est l'extension des symboles graphiques des composants)

Créer maintenant un nouveau schéma : New- Block Diagram/Schematic File
Créer un nouveau schéma nommé « ouex.bdf » avec le nouveau composant « ex1_pem».

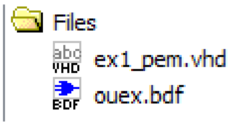


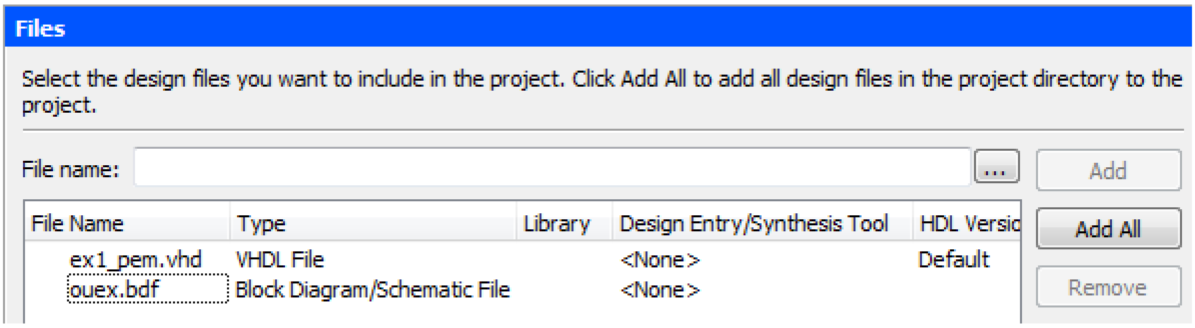


Double-cliquer sur le composant ex1_pem et indiquer comme description ex1_pem.vhd
Modifier les fichiers du projet en supprimant le schéma ex1_pem.bdf et en le remplaçant par sa description VHDL, crée précédemment, ex1_pem.vhd. Pour cela effectuer un clic-droit sur le mot file dans le gestionnaire de projet. Puis « Add-Remove files »



Naviguer pour sélectionner ex1_pem.vhd, puis Add puis OK.





Le fichier de base du projet ayant changé, il faut l'indiquer :

🖰 Clique-droit sur ouex.bdf et valider «Set as Top-Level entity »

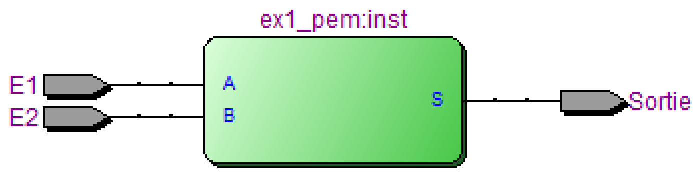
Le projet ayant une description VHDL il est possible d'effectuer une simulation fonctionnelle.
La simulation fonctionnelle (RTL) sera effectuée par rapport au design du concepteur.
La simulation temporelle (GATE) sera effectué par rapport à la compilation et à l'optimisation des phases d'analyse et de synthèse.

e. Schéma RTL

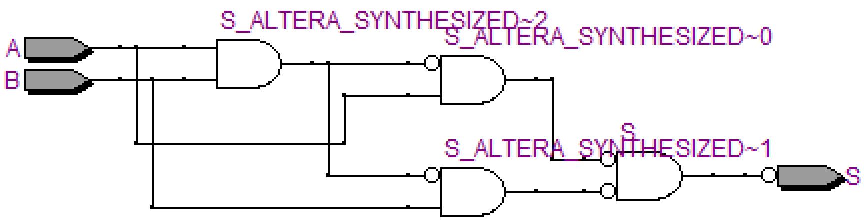
Ce schéma correspond à l'analyse RTL

🖰 Tools – Netlist Viewers - RTL Viewver.

Quartus crée le symbole correspondant à la description fonctionnelle.



🖰 🖰 sur le composant permet de voir la structure logique. (On retrouve celle du schéma), c'est cette dernière qui sera utilisée pour la simulation fonctionnelle.



f. Schéma GATE :

Il est possible de voir le schéma avant et après l'intégration dans le FPGA.

🖰 Tools – Netlist Viewers - Technologie Map Viewver (Post Fitting).

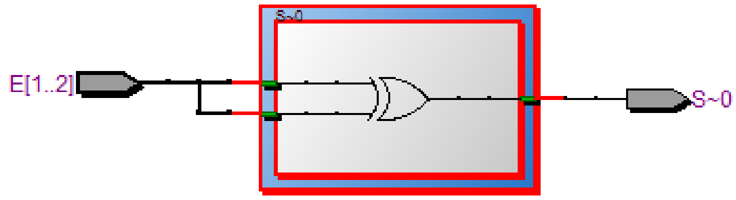
Quartus crée le symbole correspondant à la synthèse obtenue.



🖰 🖰 sur le composant permet de voir les LOGIC_CELL utilisées (ici une seule)



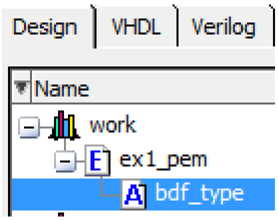
🖰 🖰 sur le LOGIC_CELL permet de voir la structure logique après optimisation (Donc un XOR)



C'est ce schéma qui sera utilisé pour la simulation temporelle.

g. *Simulation fonctionnelle*

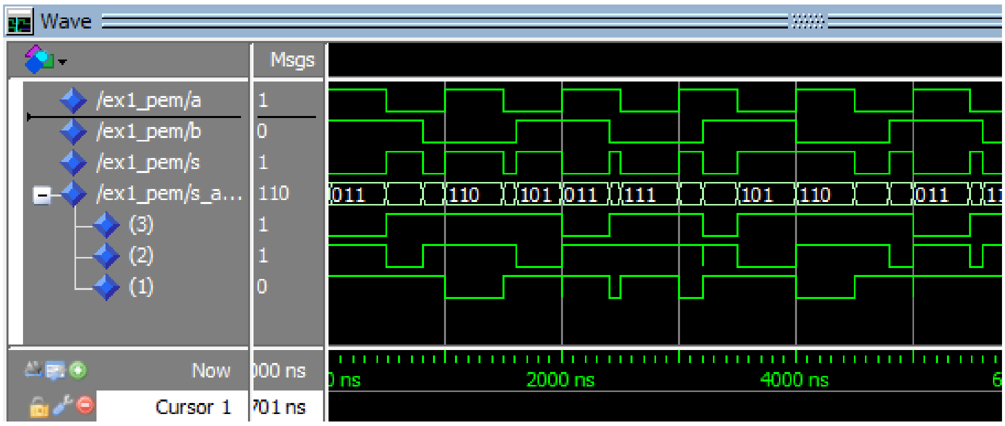
🖰 Cliquer « Tools – Run EDA simulation tools – EDA RTL Simulation “, ce qui va lancer le simulateur en mode “RTL simulation fonctionnelle”.



Procéder ensuite comme précédemment mais en choisissant cette fois ci WORK « bdf_type », et non plus GATE-WORK puis OK (pas de fichier sdf, le matériel n'étant pas simulé)

On retrouve les trois signaux a,b,s, ainsi que les trois signaux internes du schéma qui possèdent les même noms que dans la description VHDL.

Effectuer comme précédemment une simulation de 10uS en plaçant sur 'a' une horloge de période 1uS et sur 'b' une période de 1.6uS.



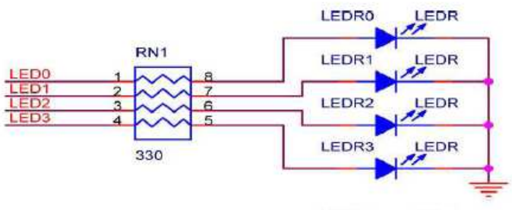
Quitter ModelSim

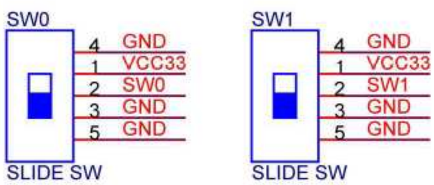
h. *Essai sur cible (KIT DE2)*

Le KIT DE2 est équipé d'interrupteurs (switchs) et de LED qui vont permettre de tester en réel le projet.

Les schémas de la carte DE2 se trouvent dans le fichier DE2_scheamtics.pdf.

Le fichier DE2_UserManuall.pdf contient des exemples et un tableau des entrées et sorties de la carte DE2. <u>Pour les essais les liaisons internes doivent être les suivantes :</u>





Signal	Interface	Brochage sur FPGA KIT DE2
a	SW0	PIN N25
b	SW1	PIN N26
s	LEDR0	PIN AE23

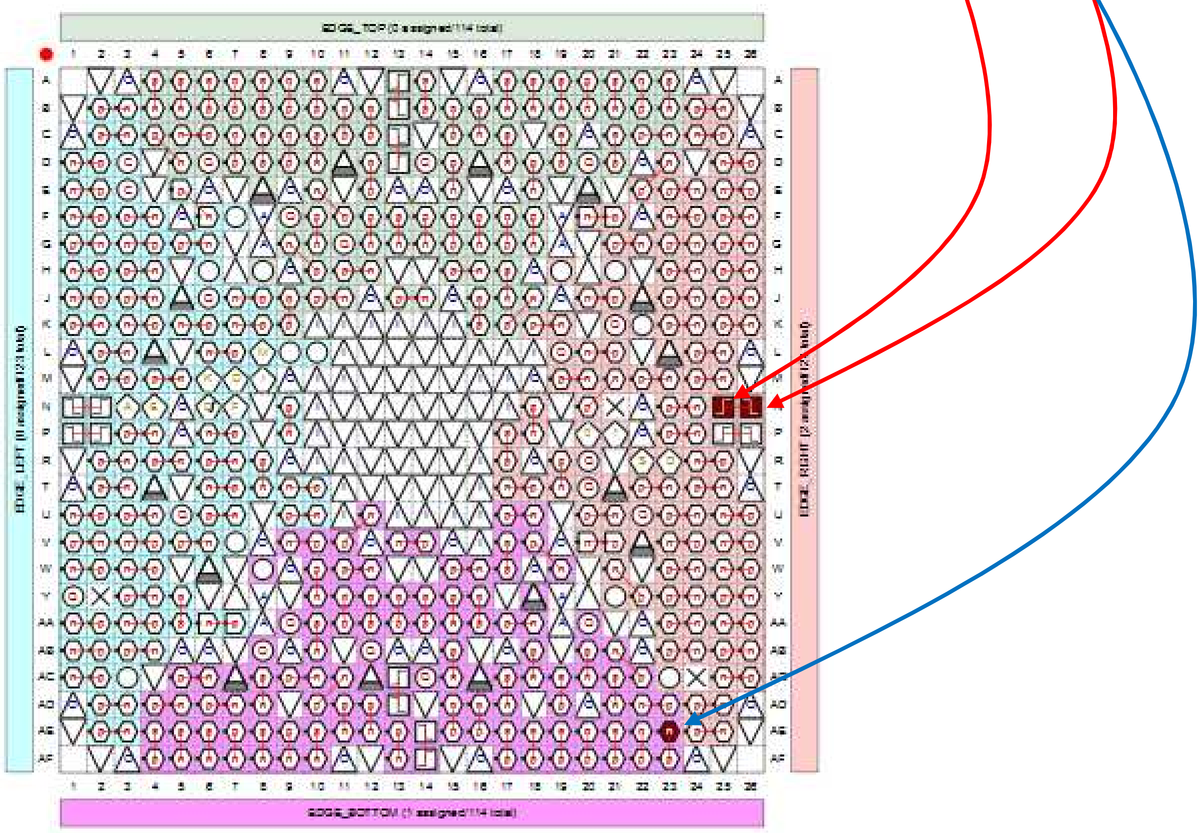
L'outil « PIN PLANER » permet d'attribuer des broches physiques à des entrées/sorties

Cliquer « Assignment – Pin Planer »
Pin Planer affiche un plan de câblage du FPGA et permet l'assignement des entrées/sorties
Configurer le brochage de a, b, s comme suit : (glisser – déposer)

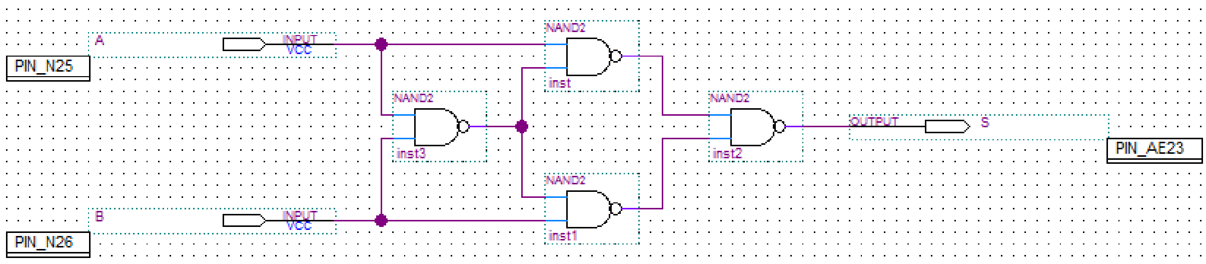
Node Name	Direction	Location	I/O Standard
A	Input	PIN_N25	3.3-V LV...default)
B	Input	PIN_N26	3.3-V LV...default)
S	Output	PIN_AE23	3.3-V LV...default)

Les broches a,b,s apparaissent maintenant sur le plan du FPGA

Cyclone II - EP2C35F672C6

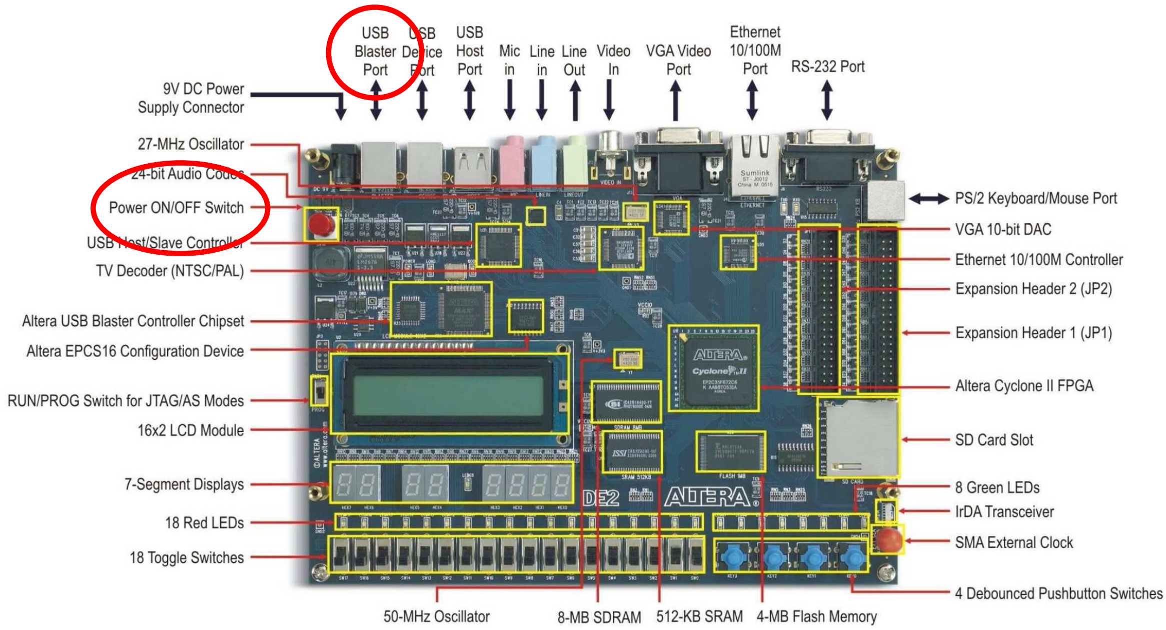


Fermer PIN Planer, les numéros de broches apparaissent sur le schéma.



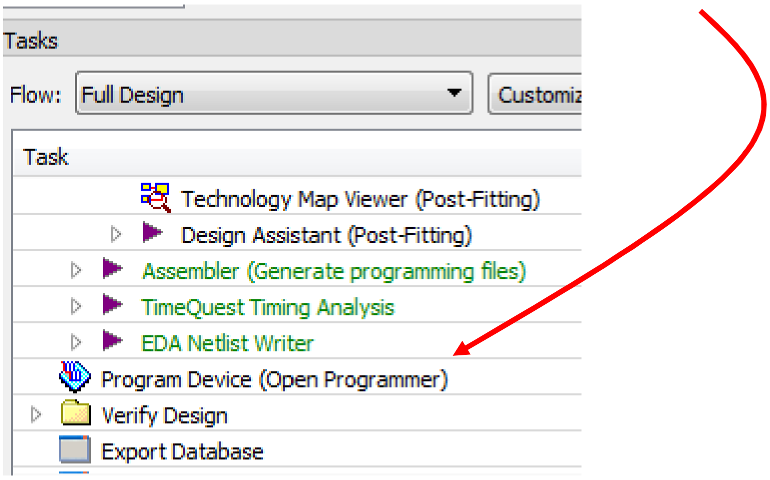
<u>Recompiler le projet.</u>

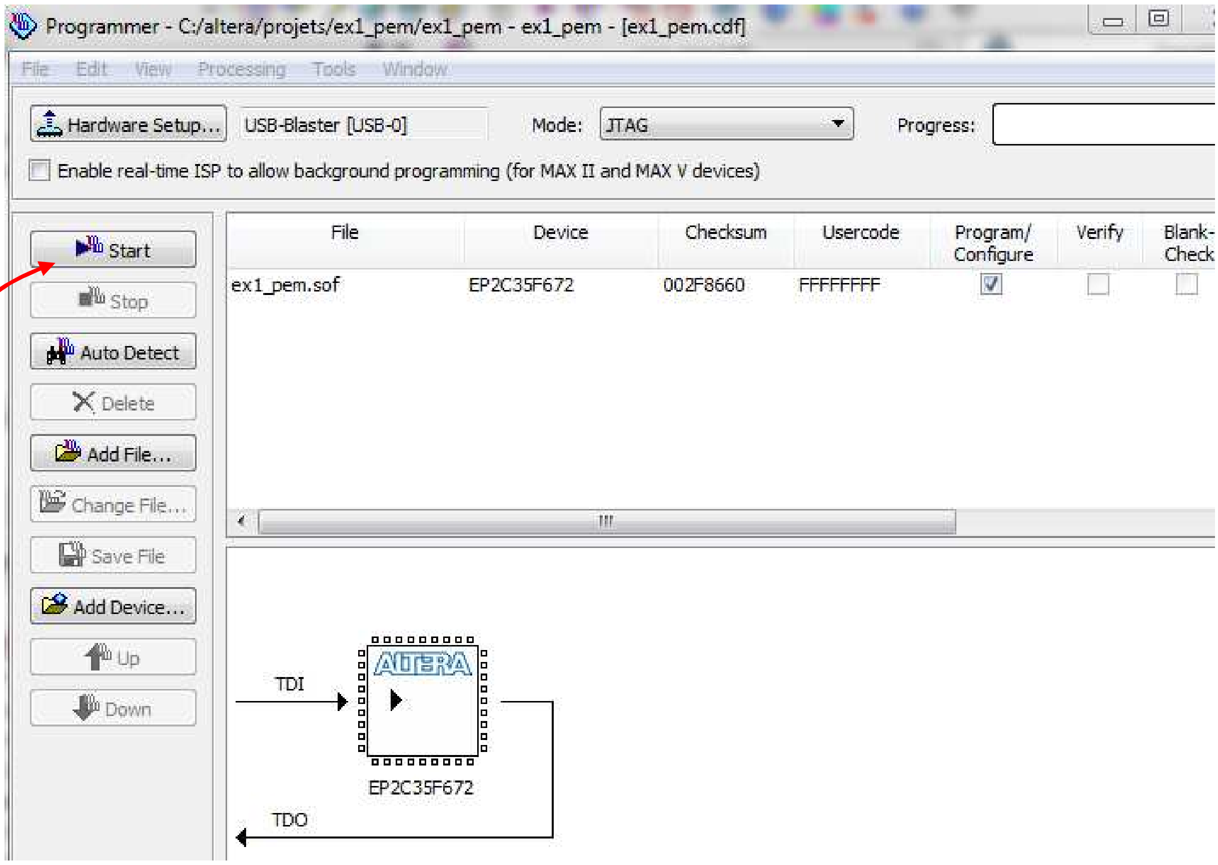
Placer un câble entre l'USB Blaster Port du KIT DE2 sur un port USB du PC.



Alimenter le KIT et appuyer sur le bouton ON/OFF, *(L'application par défaut du KIT teste les LEDs et les afficheurs).*

Sur QUARTUS, dans la fenêtre Tasks, double-clique « Program Device »





Normalement le programmeur « USB-Blaster » est automatiquement détecté.

Dans le cas contraire, cliquer sur « Hardware Setup » selectionner « USB-Blaster » puis Add Hardware.

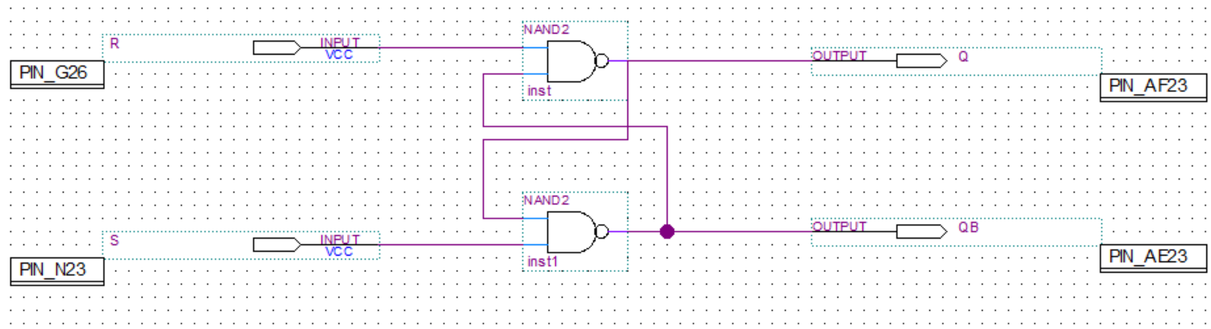
Cliquer Start, le FPGA est programmé. **Tester alors la fonction XOR sur les interrupteurs SW0, SW1 et la LED0**

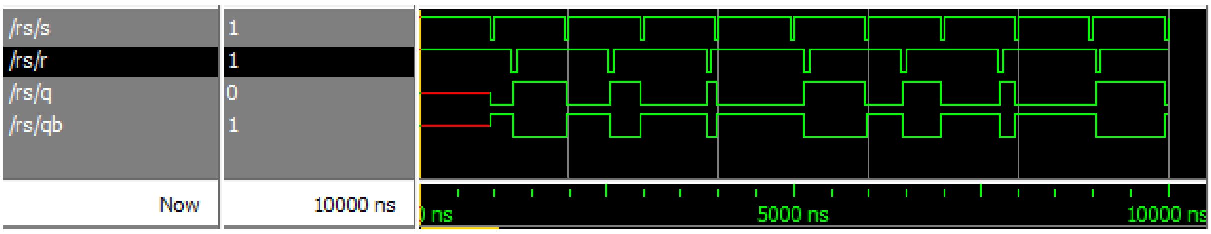




i. Exercice 2, bascule RS

De la même manière, réaliser le projet, le schéma, la simulation et les essais d'une bascule RS.



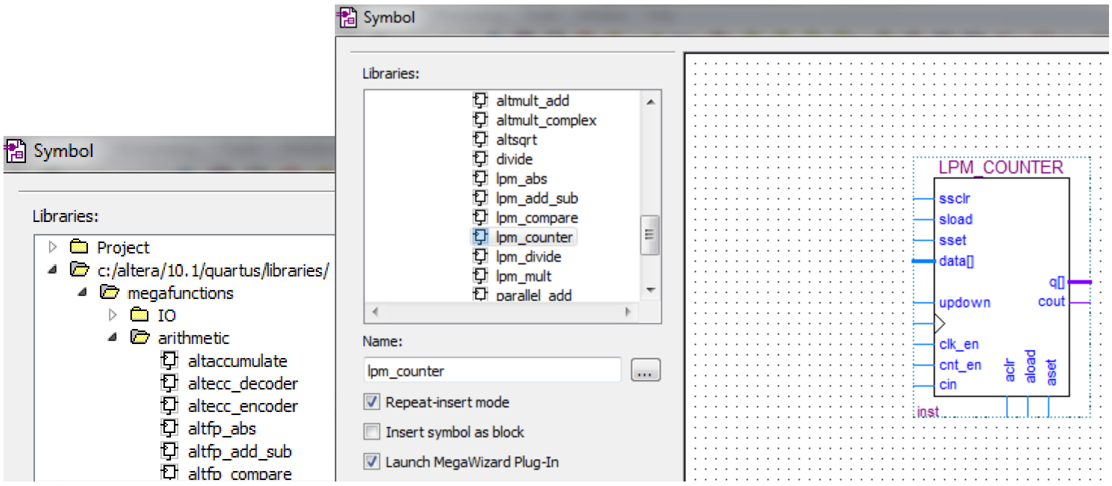


2) Megafonctions

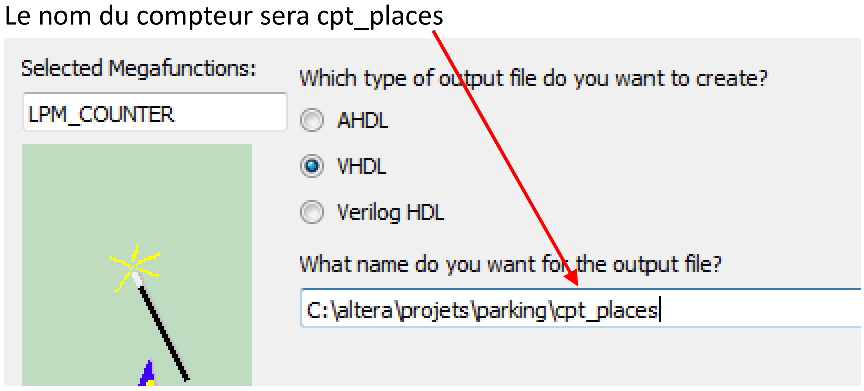
Le gestionnaire de parking met en œuvre un compteur/décompteur 12bits pour le comptage des véhicules et une fonction monostable produisant une impulsion calibrée de 1s pour la commande de la barrière.

Ces fonctions sont réalisées à partir des megafonctions de QUARTUS.

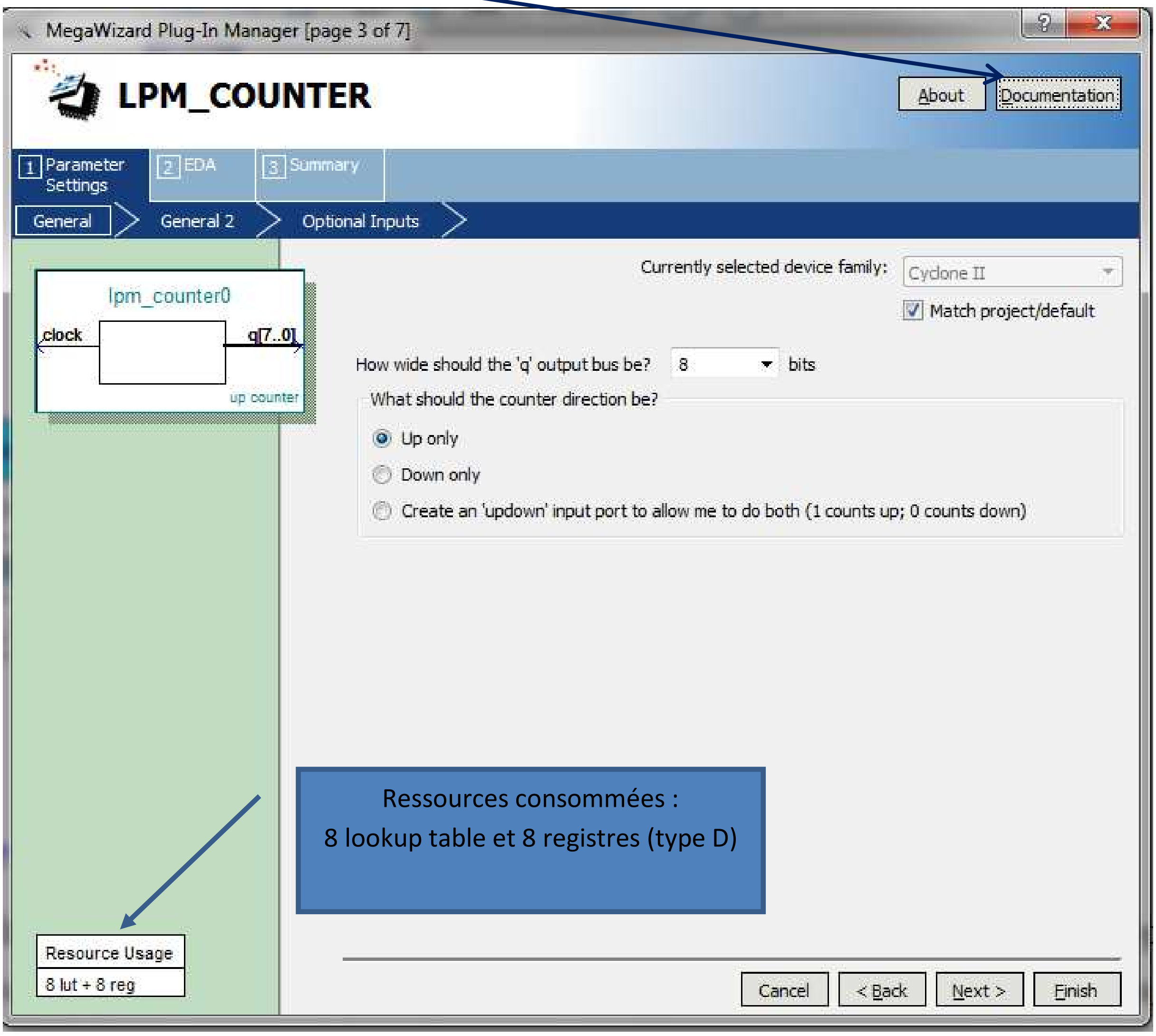
- Créer un nouveau projet « parking » destiné au KIT DE2. Créer un schéma.
- Ajouter un composant, libraires-megafonctions-arithmetic –lpm_counter



Le nom du compteur sera cpt_places



Une documentation très complète est disponible sur le site WEB d'ALTERA pour chaque megafonction.



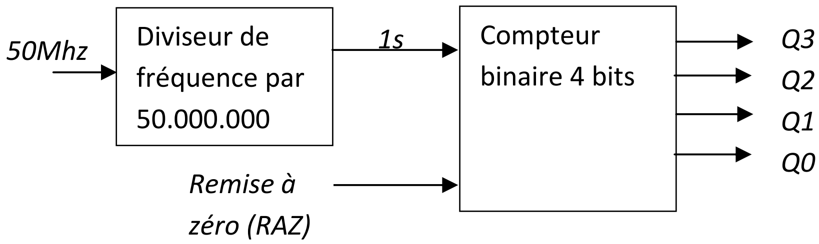
Configurer le compteur comme suit :
- 12bits, avec entrée Up/down
- modulus 1200 avec Clock Enable
- Set asynchrone, Set to 10 (pour les essais, le nombre max de véhicules sera limité à 10, il pourra être changé par la suite).

Cliquer next/ next/ finish, le composant est terminé et peut être placé sur le schéma. Pour le modifier il suffit de double-cliquer dessus.

Refermer le projet « parking » il sera complété plus tard.

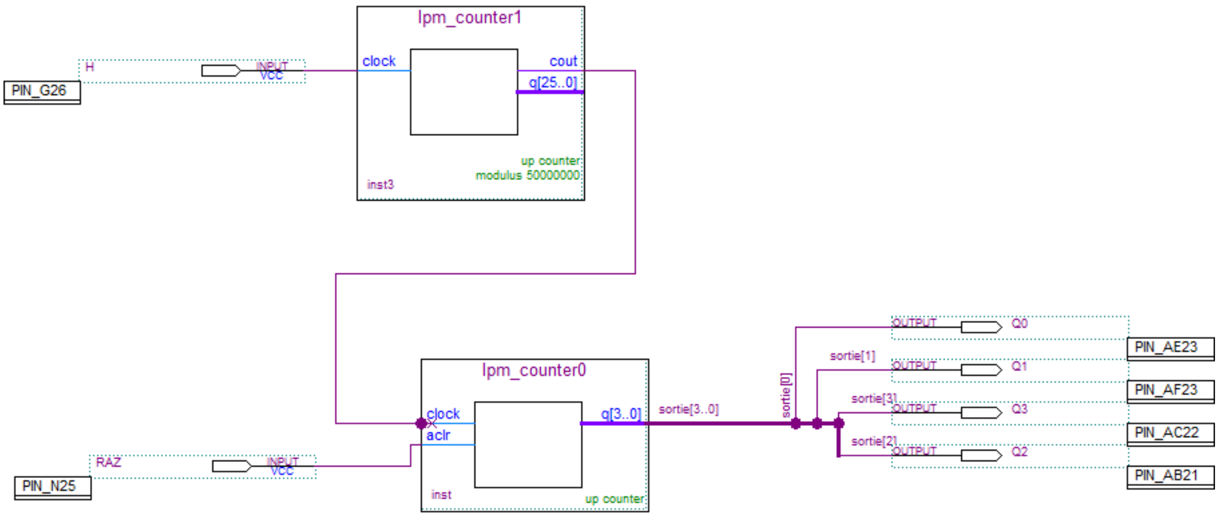
a. *Exercice 3 : compteur de secondes binaire 4 bits*

En utilisant les megafonctions, créer un projet « compteur4bits » permettant de visualiser le comptage des secondes en binaire sur LED4, LED3, LED2, LED0.
Le KIT DE2 possède un oscillateur 50MHz connecté à la broche PIN_G26 du FPGA, en divisant cette fréquence par 50.10^6 on obtient une base de temps d'une seconde soit sur sa sortie de poids le plus fort soit sur sa sortie de débordement (carry out ou cout).



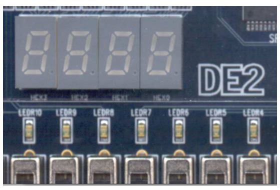
Les sorties du compteur 4 bits sont représentées par un bus.
Pour nommer un bus : sélection, clic-droit. Donner un nom au bus suivi de sa dimension entre crochets. Ex ici : sortie[3..0], les fils du bus s'appellent sortie[3], sortie[2], sortie[1], sortie[0].

Réaliser et tester sur le KIT DE2 le projet ci-dessous.

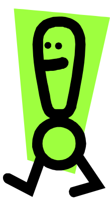


b. *Exercice 4 compteur 7 segments*

Le projet réalise un compteur binaire 16 bits visualisé sur les quatre afficheurs 7 segments de droite du KIT DE2. La période de comptage sera de 250ms

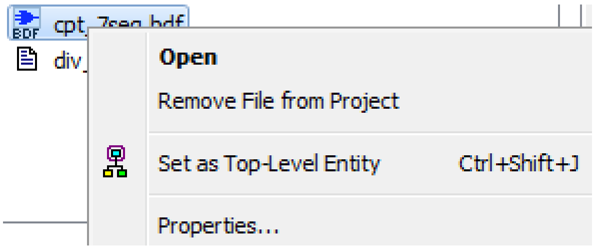


Les sorties d'un compteur étant en binaire, il est nécessaire de disposer d'une fonction « décodeur binaire/ 7- segments ». Cette fonction sera ici écrite en langage VHDL. (le langage VHDL sera étudié ultérieurement)



<u>Remarque importante :</u> L'horloge du compteur est ici asynchrone. Altera recommande de développer une logique entièrement synchronisée sur l'horloge unique du composant (ici H). Cela peut se faire en ajoutant une porte « ET » dont les entrées sont connectées à 'H' et à la sortie du diviseur et la sortie au compteur.

Créer un projet « cpt_7seg » pour le KIT DE2.
Fichier – new – Block Diagram/schematic file, enregistrer ce schéma (pour l'instant vide) sous cpt_7seg.bdf. S'assurer qu'il est bien le fichier principal du projet :
Dans « project navigator », selectionner l'onglet « Files » , puis cliquer-droit sur cpt_7seg.bdf et valider « Set as Top-Level Entity »



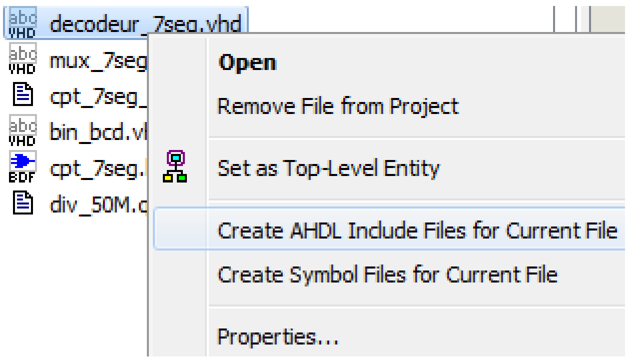
Pour créer le décodeur 7 segments on utilise le fichier VHDL suivant :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

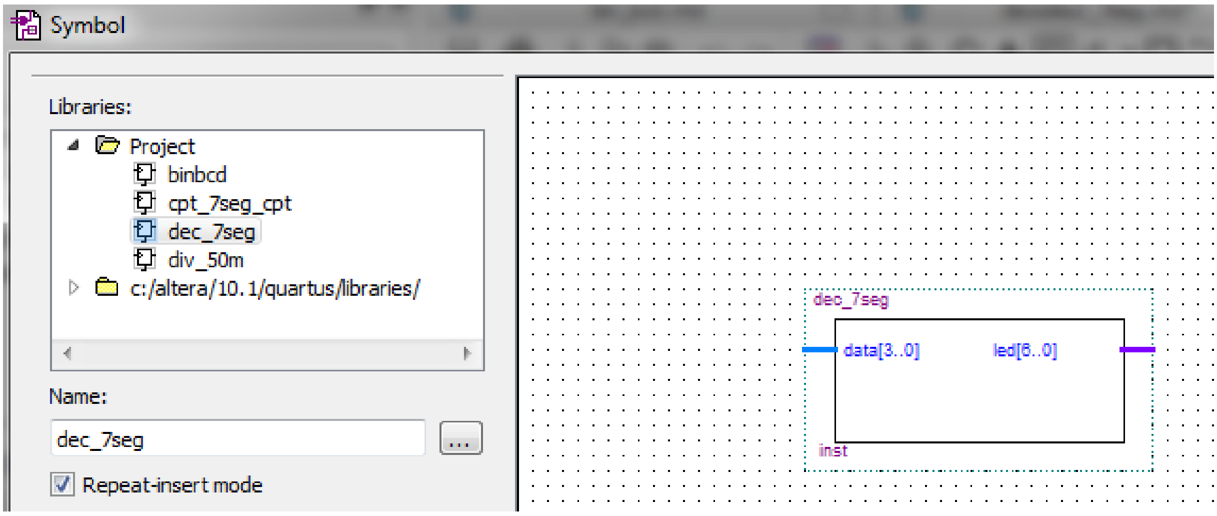
entity dec_7seg is
    Port ( data : in  STD_LOGIC_VECTOR (3 downto 0);
           led : out  STD_LOGIC_VECTOR (6 downto 0));
end dec_7seg;

architecture comportement of dec_7seg is
begin
       with data select
       led <=
       "1000000" when x"0" ,
       "1111001" when x"1" ,
       "0100100" when x"2" ,
       "0110000" when x"3" ,
       "0011001" when x"4" ,
       "0010010" when x"5" ,
       "0000010" when x"6" ,
       "1111000" when x"7" ,
       "0000000" when x"8" ,
       "0010000" when x"9" ,
       "0001000" when x"A" ,
       "0000011" when x"B" ,
       "1000110" when x"C" ,
       "0100001" when x"D" ,
       "0000110" when x"E" ,
       "0001110" when others;
end comportement;
```

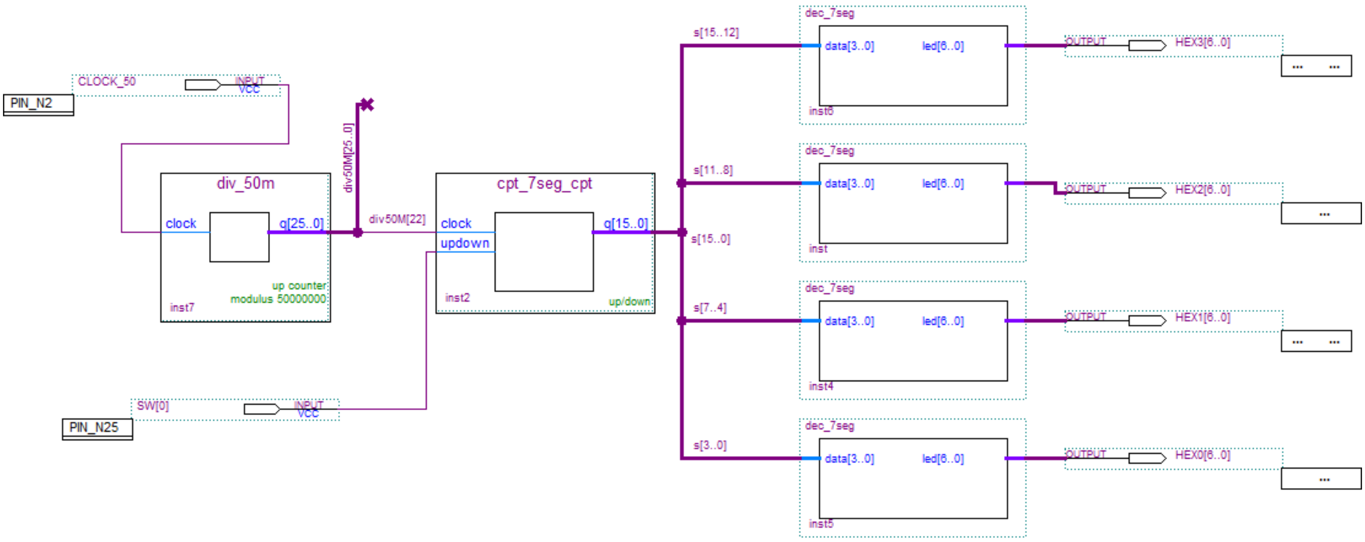
File – New – « VHDL file »
Recopier le code VHDL ci-dessus, enregistrer le fichier sous le nom « decodeur_7seg.vhd ».
Le fichier ainsi crée doit apparaitre dans « project navigator -, onglet files », sinon l'ajouter (cliquer-droit sur le mot Files, puis « add/remove files in project ».
Cliquer-droit sur « decodeur_7seg.vhd » puis « Create Symbol Files for Current File ». **QUARTUS crée un symbole graphique correspondant au code VHDL.**



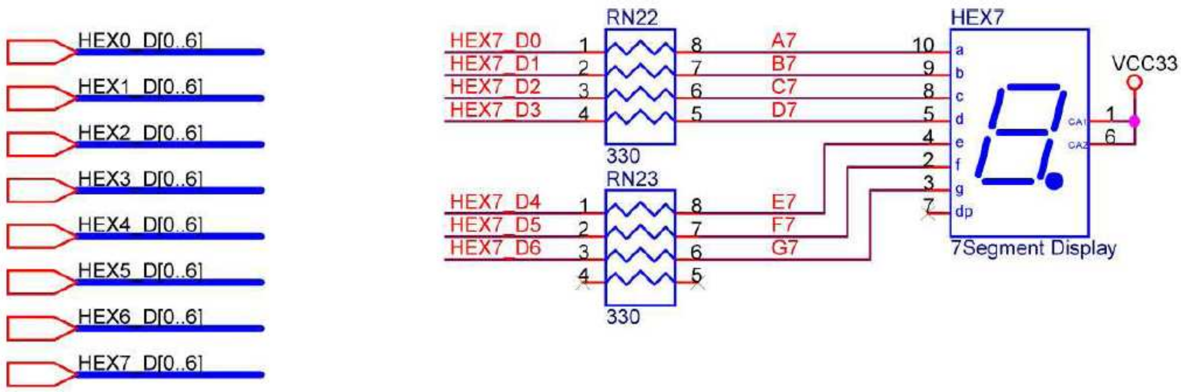
Ce symbole est maintenant accessible dans le gestionnaire de composants de l'éditeur de schéma.



Réaliser le schéma suivant <u>sans nommer les entrées/sorties</u>:



Les afficheurs du KIT DE2 sont à anodes communes et cablés chacun sur un bus comme suit :



L'édition des connexions de 7 fois 4 afficheurs sera très longue et fastidieuse sur l'éditeur de schéma de QUARTUS.

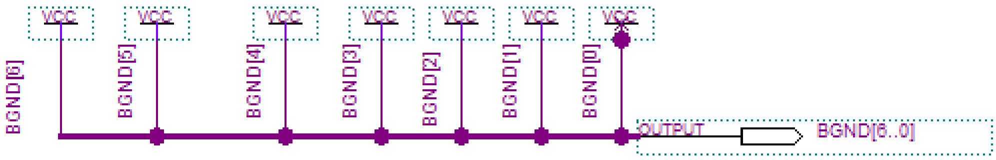
Le fichier *DE2_pin_assignments.csv* fourni sur le CDROM du KIT DE2 contient la description du cablage de la carte DE2, il peut être importé dans QUARTUS, les broches du FPGA du KIT DE2 seront alors nommées automatiquement.

Assignments – Imports Assignments – selectionner *DE2_pin_assignments.csv*. (le fichier est à récupérer sur le CD accompagnant la carte DE2)
http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=53&No=30&PartNo=4

Assignement – Assignment editor ouvre une fenetre permettant de voir les assignements de broches.
Les interrupteurs s'appellent SW[0], SW[1] etc… les LEDs rouges LEDR[0], LEDR[1] etc… L'horloge 50MHz CLOCK_50…
Les afficheurs 7 segments sont nommés HEX0, HEX1, HEX 2…, et sont connectés à des bus HEX0[6..0], HEX1[6..0], HEX2[6..0], ….
Les segments « a à g » des afficheurs sont respectivement les bits 0 à 6 des bus.
Nommer les connecteurs du schéma comme suit :
- L'horloge 50 MHz : CLOCK_50
- Le comptage/decomptage : SW[0]
- Les bus des afficheurs : HEX0[6..0] pour l'afficheur du bas du schéma jusqu'à HEX3[6..0] pour l'afficheur du haut.

Compiler et essayer le compteur sur le KIT DE2.

Pour éteindre les afficheurs inutiles (les quatre de gauche), il suffit de polariser les cathodes avec VCC.
Créer un nouveau schéma comme suit , nommer-le efface_aff7seg (VCC se trouve dans la catégorie « other »):



A partir de ce schéma créer un nouveau composant (File-Create/Update-Create symbol File for current file)
Ajouter quatre sorties bus pour les quatre afficheurs manquant sur le schéma (HEX4, HEX5,HEX6,HEX7) et les connecter à quatre composants efface_aff7seg.

c. *Exercice 5 : Horloge temps réel*

Réaliser une horloge temps réel (Heure, minute, seconde) sur les six afficheurs de gauche du KIT DE2. Un bouton permettra une incrémentation rapide des heures et un autre des minutes pour permettre la mise à l'heure.
L'affichage étant en décimal, il est necessaire de disposer d'un décodeur binaire/décimal.

Ci dessous le code VHDL du décodeur binaire/décimal
Pour en savoir plus sur le VHDL, consulter le livre gratuit de J.Weber et S.Moutault

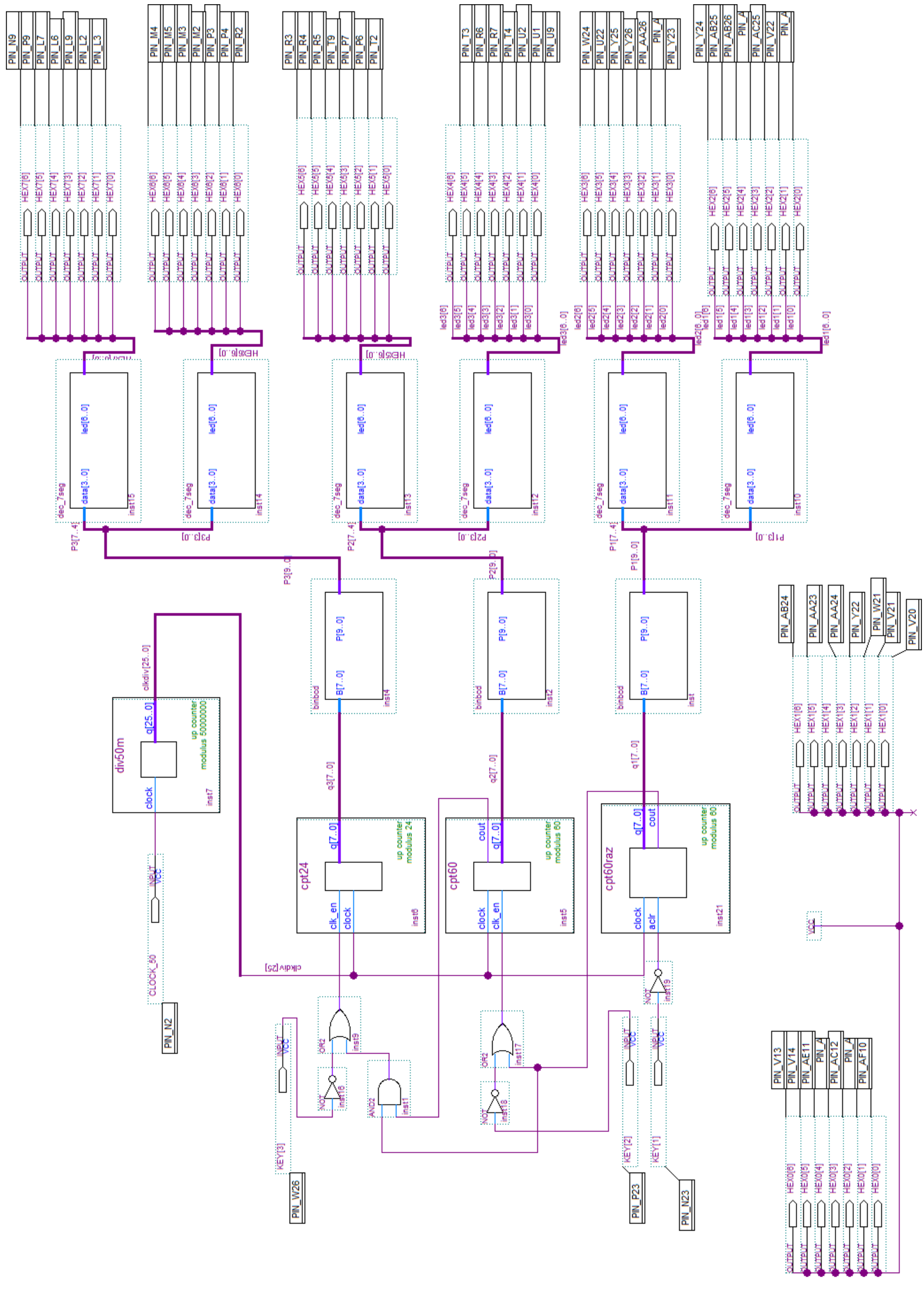
http://books.google.fr/books?id=AKolOwjcqnUC

```
-- Convertisseur BIN/BCD 8 bits
-- Convertit un nombre binaire sur 8 bits en trois chiffres decimaux
-- (12 bits)
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity binbcd is
            port (
            B: in STD_LOGIC_VECTOR (7 downto 0);
-- nombre en entrée sur 8 bits
            P: out STD_LOGIC_VECTOR (11 downto 0)
-- nombre decimal en sortie sur 12 bits
            );
end binbcd;

architecture binbcd_arch of binbcd is
      begin
      bcd1: process(B)
            variable z: STD_LOGIC_VECTOR (19 downto 0);
            begin
                  for i in 0 to 17 loop
                  z(i) := '0';
            end loop;
            z(10 downto 3) := B;
            for i in 0 to 4 loop
            if z(11 downto 8) > 4 then
                  z(11 downto 8) := z(11 downto 8) + 3;
            end if;
            if z(15 downto 12) > 4 then
                  z(15 downto 12) := z(15 downto 12) + 3;
            end if;
            z(17 downto 1) := z(16 downto 0);
            end loop;
            P <= z(19 downto 8);
      end process bcd1;
end binbcd_arch;
```

Schéma complet de l'horloge temps réel



Annexe : principales extensions de fichiers

fichier descriptif du projet : **.qpf**

Fichiers de description

description graphique : **.bdf**
description vhdl : **.vhd**
description verilog : **.vo**

Fichiers de programmation

composants EEPROM : **.pof**
composants SRAM : **.sof**

Fichiers divers

rapport de compilation : **.rpt**
assignation des broches : **.acf**
symbole graphique d'une description : **.sym**